IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2014–04–16.
Submitted on 2013–11–04, accepted on 2014–01–31.
ISSN 2105–1232 © 2014 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2014.107

Multiscale Retinex

Ana Belén Petro[1], Catalina Sbert[2], Jean-Michel Morel[3]

[1] University of Balearic Islands (Spain) (anabelen.petro@uib.es)
[2] University of Balearic Islands (Spain) (catalina.sbert@uib.es)
[3] ENS Cachan (France) (morel@cmla.ens-cachan.fr)

Abstract

While the retinex theory aimed at explaining human color perception, its derivations have led to efficient algorithms enhancing local image contrast, thus permitting among other features, to "see in the shadows". Among these derived algorithms, Multiscale Retinex is probably the most successful center-surround image filter. In this paper, we offer an analysis and implementation of Multiscale Retinex. We point out and resolve some ambiguities of the method. In particular, we show that the important color correction final step of the method can be seriously improved. This analysis permits to formulate an automatic implementation of Multiscale Retinex which is as faithful as possible to the one described in the original paper. Overall, this implementation delivers excellent results and confirms the validity of Multiscale Retinex for image color restoration and contrast enhancement. Nevertheless, while the method parameters can be fixed, we show that a crucial choice must be left to the user, depending on the lightning condition of the image: the method must either be applied to each color independently if a color balance is required, or to the luminance only if the goal is to achieve local contrast enhancement. Thus, we propose two slightly different algorithms to deal with both cases.

Source Code

The source code, the code documentation, and the online demo are accessible at the IPOL web page of this article[1]. In this link an implementation is available for download. Compilation and usage instruction are included in the README.txt file of the compressed archive. This software includes the implementations of algorithms potentially linkable to patents.

Keywords: Retinex theory; color restoration; contrast enhancement

[1] http://dx.doi.org/10.5201/ipol.2014.107

1 Overview

The retinex theory was developed by Land and McCann [13] to model how the human visual system perceives a scene. They established that the visual system does not perceive an absolute lightness but rather a relative lightness, namely the variations of lightness in local image regions. In this first work, Land and McCann proposed a complex algorithm involving image paths to compute the relative lightness. Many different implementations and interpretations have followed this first work. They can be divided into two major groups. The first group explores the image relative lightness using a variety of image paths or comparing the current pixel color to a set of random pixels [10, 11, 5, 16, 21]. The second group uses a convolution mask or variational techniques to compute a local contrast result [11, 8, 7, 9, 1, 18, 15, 2].

The original retinex algorithm, proposed by Land [13], considers all possible paths starting at random points and ending at the pixel at which the lightness value is computed. This lightness is then defined as the average of the products of ratios between the intensity values of subsequent edge points in the path. In order to remove the effects of nonuniform illumination over the scene, the ratio is considered unitary if it does not differ from 1 by more than a fixed threshold value. Passing to formulas, we consider a collection of N paths $\gamma_1, \ldots, \gamma_k, \ldots, \gamma_N$ starting at x and ending at an arbitrary image pixel y_k. Let n_k be the number of pixels of the path γ_k, and denote by $x_{t_k} = \gamma_k(t_k)$ for $t_k = 1, \ldots, n_k$ and by $x_{t_k+1} = \gamma_k(t_k + 1)$ the subsequent pixel of the path so that $\gamma_k(1) = x$ and $\gamma_k(n_k) = y_k$. Then, the lightness $L(x)$ of a pixel x in a given chromatic channel is the average of the relative lightness at x over all paths, that is

$$L(x) = \frac{\sum_{k=1}^{N} L(x; y_k)}{N} \,, \tag{1}$$

where $L(x; y_k)$ denotes the relative lightness of a pixel x with respect to y_k on the path γ_k defined by

$$L(x; y_k) = \sum_{t_k=1}^{n_k} \delta \left[\log \frac{I(x_{t_k})}{I(x_{t_k+1})} \right] \,, \tag{2}$$

and, for a fixed contrast threshold t,

$$\delta(s) = \begin{cases} s & \text{if } |s| > t \\ 0 & \text{if } |s| < t \end{cases} . \tag{3}$$

The basic Retinex has a reset mechanism, which is described by Land [10] as the heart of the Retinex model to find the highest reflectance in the path. This reset mechanism consist of forcing the cumulated relative lightness to be zero if a lighter area is found. The reset ensures that all paths start from regions where the maximal luminance value is attained. The Retinex algorithm with both the threshold and reset mechanisms has been written with a mathematical notation and then analyzed by Provenzi et al. in [20].

In a posterior work Land [12], published an alternative to the initial random walk algorithm. In practical terms, this new implementation computes the lightness as the ratio between the value of a pixel and the average value of the surrounding samples, considering these surrounding samples to have density proportional to the inverse of the square distance. We can consider this operation, which is a sort of high-pass filter, as a center/surround operation. Taking for example a Gaussian filter G_σ, the operation amounts to set $L(x) := \frac{I(x)}{(I * G_\sigma)(x)}$, which implies

$$\log L(x) := \log I(x) - \log(I * G_\sigma)(x). \tag{4}$$

By changing the position of the logarithm in the above formula and setting

$$\log L(x) := \log I(x) - (\log I) * G_\sigma(x),$$

we would obtain a classic high pass linear filter, but applied to $\log I$ instead of I.

Jobson et al. [8] explored the properties of the center/surround function, for defining a specific form for the center/ surround retinex. They investigated the fundamental issues:

- placement of the log function;
- functional form of the surround;
- scale of the surround filter;
- treatment of the retinex outputs prior to final display.

This analysis led to the so-called single-scale retinex method constructed on formula (4), which has the property of controlling a trade-off between rendition and dynamic range compression. In a posterior paper [7], the same authors extended the work to multiple scales, and proposed the by now classic Multiscale Retinex, which is the object of the present paper. In this work, we attempt at an almost faithful implementation of the original Multiscale Retinex. We shall only modify the post-processing steps, for which we show that the original version induces unacceptable failures. We shall replace it by a simpler technique achieving the same goals. We shall also define two different versions of the method, depending on whether the image requires a color balance or not.

The plan of the paper is as follows. Section 2 is devoted to a precise description of the original Multiscale Retinex for restoring image color. In Section 3, we point out the implementation problems that arise, and explain how we have worked them out. In this section we also discuss the final gain/offset step, which transforms the logarithmic domain into the display domain. In Section 4 we detail our implementation of the algorithm. Section 5 gives implementation details on the the Gaussian convolutions. The online demo is detailed in Section 6. Finally, Section 7 analyzes some results of the algorithm and discusses their practical use and application.

2 Original Multiscale Retinex Algorithm

2.1 Single-Scale Retinex

Single-Scale Retinex is a member of the class of center/surround functions where the output is determined by the difference between the input value (center) and an average of its neighborhood (surround). The general mathematical form of a Single-Scale Retinex (SSR) is, extending (4) to all channels, given by

$$R_i(x,y) = \log(I_i(x,y)) - \log(I_i(x,y) * F(x,y)) \tag{5}$$

where I_i is the input image on the i-th color channel, R_i is the retinex output image on the i-th channel and F is the normalized surround function. This operation is performed on each color channel. Equation (5) implies that a "color constancy" property is achieved by the algorithm. Indeed, taking the classic assumption that

$$I_i(x,y) = S_i(x,y)r_i(x,y)$$

where S_i is the illumination and r_i is the scene reflectance, then (5) rewrites

$$R_i(x,y) \simeq \log \frac{S_i(x,y)r_i(x,y)}{\overline{S_i}\overline{r_i}}$$

where the bars denote the weighted average value and S_i is assumed to vary smoothly and therefore to be locally almost constant. Thus, we can assume that $S_i \simeq \overline{S_i}$. It follows that

$$R_i \simeq \log \frac{r_i}{\overline{r_i}}$$

which therefore is illumination independent. This explains why the method should attenuate shading effects and shadows in the image. Many surround functions have been proposed. For example, Land in [12] proposed a singular radial kernel (its integral in the continuous domain is infinite), namely

$$F(x,y) = \frac{C}{x^2+y^2}.$$

Jobson et al. [8] proposed instead a Gaussian function

$$F(x,y) = C\exp[-(x^2+y^2)/2\sigma^2]$$

where σ, the filter standard deviation, controls the amount of spatial detail which is retained, and C is a normalization factor such that

$$\int F(x,y)dxdy = 1.$$

The value of σ cannot be theoretically modeled and determined. Basically there is a trade-off between enhancement of the local dynamics, (revealing for example details in the shadows), and color rendition. In [8], the authors concluded that $\sigma = 80$ is a good choice.

Another question which has not been studied theoretically is the placement of the logarithm function in (5). Some researchers [6] have placed it before and others [17] after the application of the surround convolution. Jobson et al. [8] tested both options. Mathematically, they studied the difference between the two possible outputs

$$R_1(x,y) = \log(I(x,y)) - \log(I(x,y) * F(x,y)) \quad \text{and} \quad R_2(x,y) = \log(I(x,y)) - \log(I(x,y)) * F(x,y),$$

which are of course not equivalent. R_1 is the logarithm of the ratio between the image and a weighted average of it, while R_2 is the logarithm of the ratio between the image and a weighted product. This amounts to choosing between an arithmetic mean and a geometric mean. Jobson et al. [8] confirmed with experiments that it was better to place the logarithm function after the surround.

2.2 Multiscale Retinex (MSR)

Because of the trade-off between dynamic range compression and color rendition, the choice of the right scale σ for the surround filter $F(x,y)$ is crucial in Single Scale Retinex (5). Multiscale Retinex (MSR) seems to afford an acceptable trade-off between a good local dynamic range and a good color rendition. The MSR output is defined as a weighted sum of the outputs of several SSRs. This led Jobson et al. [7] to propose the *multiscale retinex* formula

$$R_{MSR_i} = \sum_{n=1}^{N} w_n R_{n_i} = \sum_{n=1}^{N} w_n \left[\log I_i(x,y) - \log(F_n(x,y) * I_i(x,y))\right] \tag{6}$$

where N is the number of scales, w_n is the weight of each scale and $F_n(x,y) = C_n \exp[-(x^2+y^2)/2\sigma_n^2]$.

The questions that immediately arise are: what scales to choose, and how many, and what are the ideal values for the weights. According to Jobson et al. three scales are enough for most images, and the weights can be identical. These authors experimentally fixed the scales to 15, 80 and 250.

constant	N	σ_1	σ_2	σ_3	α	β	w_n	G	b
value	3	15	80	250	125	46	1/3	192	-30

Table 1: List of constants in the implementation of Multiscale Retinex [7]

2.3 The Color Restoration Step of Multiscale Retinex (MSRCR)

Certain assumptions are sometimes made about the general nature of the color components of images. One of such assumptions is the *gray world assumption* which states that given an image with sufficient amount of color variations, the average value of the red, green and blue components of the image should average out to a common gray value [3], [4]. In images which violate the gray-world assumption, that is, images where a certain color may dominate, the above retinex procedure produces grayish images by decreasing the color saturation. To solve this problem, Jobson et al. [7] proposed to complete the algorithm with a color restoration step. They proposed to modify the MSR output by multiplying it by a color restoration function of the chromaticity. The first move is to compute the chromaticity coordinates

$$I'_i(x,y) = \frac{I_i(x,y)}{\sum_{j=1}^{S} I_j(x,y)}$$

for the ith color band, where S is the number of spectral channels. (Generally, $S = 3$ for RGB color space.) The restored color MSR is given by

$$R_{MSRCR_i}(x,y) = C_i(x,y) R_{MSR_i}(x,y) \tag{7}$$

where

$$C_i(x,y) = f(I'_i(x,y))$$

is the ith band of the color restoration function (CRF). Jobson et al. [7] found that the CRF providing the best overall color restoration is defined by

$$C_i(x,y) = \beta \log[\alpha I'_i(x,y)] \tag{8}$$

where β is a gain constant, and α controls the strength of the nonlinearity. The same authors determined experimentally a single set of values for β and α that seem to work for all spectral channels (see Table 1).

2.4 Gain/Offset of MSRCR

In his original work [12] where the center/surround version of retinex is introduced, Land does not propose any solution for the treatment of the retinex output. Since the result of the above process can yield negative and positive RGB values with arbitrary bounds, the range of values has to be transformed into the display domain, $[0, 255]$. In a posterior work, Moore et al. [17] proposed an automatic treatment of the retinex output prior to display, where each color channel is adjusted by the absolute minimum and maximum of the three color bands:

$$R_{MSRCR_i}(x,y) = 255 \frac{R_{MSRCR_i}(x,y) - \min_i(\min_{(x,y)} R_{MSRCR_i}(x,y))}{\max_i(\max_{(x,y)} R_{MSRCR_i}(x,y)) - \min_i(\min_{(x,y)} R_{MSRCR_i}(x,y))}.$$

Some time later, Jobson et al. [8] observed that the particular shape of the retinex output histogram (Figure 1) justified the use of some constant parameters to compute the linear transformation between the logarithmic domain and the display domain, which does not vary from image to image. The

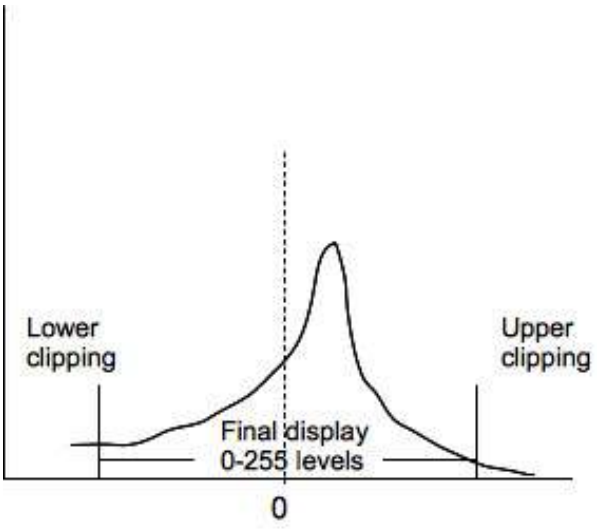

Figure 1: The histogram shape of the retinex output.

special form of the resulting image histogram, with quite large tails, shows that some clipping of the extreme color values is needed to achieve a good contrast. In this first work [8], Jobson et al. call to this process "canonical gain/offset", but they did not specify the values chosen for computing this. In a posterior paper [7], they proposed the formula

$$R_{MSRCR_i}(x,y) = G[R_{MSRCR_i}(x,y) - b],$$

where G and b are the final gain and offset values and the proposed values are represented in Table 1.

3 Implementation Problems and Our Solutions

3.1 "Canonical Gain/Offset" Problems

Our experiments with many images indicate that it is difficult to find a "canonical gain/offset" and that producing good results with the values proposed by the authors is simply not possible. We observed that the appearance of the result is affected by the gain/offset procedure. This fact had already been observed by Barnard et al. [1]. These authors proposed to clip a fixed percentage of the pixels on either side of the dynamic range. The extrema of the clipping points in each of the three channels are used to apply the same transformation to all three.

In our implementation, we adopted the Simplest Color Balance algorithm proposed by Limare et al. [14]. Simplest Color Balance simply stretches, as much as it can, the values of the three channels Red, Green, Blue (RGB), so that they occupy the maximal possible range $[0, 255]$, clipping some percentage of pixels on either side. The simplest way to do so is to apply an affine transform $ax + b$ to each channel, computing a and b so that the maximal value in the channel becomes 255 and the minimal value 0. Section 7 will illustrate the difference between several possible postprocessing procedures. Figure 2 shows an example of the result obtained when varying the filter scale in Single Scale Retinex, compared with Multiscale Retinex (MSR) and with Multiscale Retinex with color restoration (MSRCR). In all the outputs we applied a simplest color balance with 1% pixel clipping on either side of the dynamical range. We can observe that small scales produce halo artifacts and a loss of the hue, while a large scale produces a detail loss and a weak dynamic range compression.

Figure 2: Top left: original image of size 2000×1312. Top right: SSR with $\sigma = 15$. Middle left: SSR with $\sigma = 80$. Middle right: SSR with $\sigma = 250$. Bottom left: MSR. Bottom right: MSRCR.

3.2 Color Restoration Problems

We discovered that the color restoration function is at risk of inverting colors in the image. This effect can be observed in images with saturated colors. Using the constant parameters proposed by Jobson et al. (Table 1), pixels with values near 0 in some channel may jump to values near 255. Vice versa, values near 255 may go to 0. An example can be observed in Figure 3 showing a blue ball in the top left corner. For some pixels in this ball, the red channel of the input image is zero. For these pixels the red channel of the Multiscale Retinex output is negative and the CRF function at these pixels turns out to be also negative. Thus, the red channel of the Multiscale Retinex with color restoration for these pixels becomes positive and their values are changed by the postprocessing step into a value higher than the image average. This explains why in Figure 3 right the blue ball becomes magenta in some pixels. Faced with the problem that the colors change in an unpredictable way, one might be

Figure 3: Left: original image. Middle: MSR without color restoration. Right: MSRCR

tempted to eliminate the color restoration step. But without this color restoration step the images become grayish. Indeed, each pixel is compared to the average of its surrounding neighborhood and only the difference with this average is kept. In that way the difference can become small even in regions which had a strong hue, and the output color becomes grayish. Jobson et al. [7] therefore proposed a color restoration step to recover some of the lost color.

A good solution to this problem is to apply MSR on the intensity channel. Then, the output colors are computed so that the chromaticity is the same as in the original image. Working on the intensity channel colors are better preserved. Define the intensity channel as $I = \frac{\sum_{j=1}^{S} I_j}{S}$, where S is the number of channels. This version of MSR applies Formula (6) to I, obtaining a result R_I. Then a linear transformation is applied to the intensity output to stretch the result to $[0, 255]$. Finally, considering this final output intensity, the color channels are deduced by maintaining their initial chromaticity by

$$R_i = I_i \frac{R_I}{I}.$$

Figure 4 shows the difference between the results obtained when applying MSRCR to each color channel, and the results obtained by working on the intensity channel and maintaining the chromaticity.

Figure 4: Left: MSRCR on color channels. Right: MSR on intensity channel.

4 Our algorithm

The previous considerations lead to propose two possible algorithms as sound MSRCR implementations: a version applied to the three color channels with color restoration and faithful to the Jobson

et al. work, which we called properly MSRCR, and a version applied to the intensity channel without color restoration which we can call MSR with chromaticity preservation (MSRCP). Next, we detail the first version.

Algorithm 1: MSRCR algorithm

Data: I input color image; $\sigma_1, \sigma_2, \sigma_3$ the scales; s_1, s_2 the percentage of clipping pixels on each side
Result: Output color image
begin
  foreach $c \in \{R, G, B\}$ **do** ▷ For each color channel
    foreach σ_i **do** ▷ For each scale
      $\text{Diff}_{i,c} = \log(I_c) - \log(I_c * G_{\sigma_i})$ ▷ Single Scale Retinex
    end
    $\text{MSR}_c = \sum_i \frac{1}{3}\text{Diff}_{i,c}$ ▷ MultiScale Retinex
    $\text{MSRCR}_c = \text{MSR}_c \times (\log(125 I_c) - \log(I_R + I_G + I_B))$ ▷ Color Restoration
    $\text{Out}_c = \text{SimplestColorBalance}(\text{MSRCR}_c,\ s_1,\ s_2)$
  end
end

The second proposed version is

Algorithm 2: MSRCP algorithm

Data: I input color image; $\sigma_1, \sigma_2, \sigma_3$ the scales; s_1, s_2 the percentage of clipping pixels on each side
Result: MSRCP output color image
begin
  $\text{Int} = (I_R + I_G + I_B)/3$ ▷ Compute the intensity channel
  foreach σ_i **do** ▷ For each scale
    $\text{Diff}_i = \log(\text{Int}) - \log(\text{Int} * G_{\sigma_i})$ ▷ Single Scale Retinex
  end
  $\text{MSR} = \sum_i \frac{1}{3}\text{Diff}_i$ ▷ MultiScale Retinex
  $\text{Int}_1 = \text{SimplestColorBalance}(\text{MSR},\ s_1,\ s_2)$
  foreach *pixel* i **do**
    $B = \max(I_R[i], I_G[i], I_B[i])$
    $A = \min\left(\frac{255}{B}, \frac{\text{Int}_1[i]}{\text{Int}[i]}\right)$ ▷ Compute the amplification factor
    $\text{MSRCP}_R[i] = A \cdot I_R[i]$ ▷ Compute each color channel
    $\text{MSRCP}_G[i] = A \cdot I_G[i]$
    $\text{MSRCP}_B[i] = A \cdot I_B[i]$
  end
end

5 Implementation of the Convolution

MSR requires convolving the image with Gaussian functions. For large values of σ, this operation in the spatial domain is extremely time consuming. This can be solved by working in the Fourier domain where the convolution becomes a mere multiplication. In fact, the convolution with a normalized Gaussian can be implemented exactly in the discrete Fourier domain as shown in Morel and Yu [19]. We detail the algorithm in the next section.

5.1 The Discrete Fourier Transform

Denote a digital image by $(u_{m,n})$ with $0 \leq m \leq M-1$, $0 \leq n \leq N-1$.
Definition We define the *Discrete Fourier Transform (DFT)* of a digital image $(u_{m,n})_{m,n}$ by

$$\widehat{u}_{k,l} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} u_{m,n} \exp\left(-\frac{2i\pi mk}{M}\right) \exp\left(-\frac{2i\pi nl}{N}\right) \tag{9}$$

for all $0 \leq k \leq M-1$ and $0 \leq l \leq N-1$.
Definition We define the *Inverse Discrete Fourier Transform (IDFT)* of a $(\widehat{u}_{k,l})_{k,l}$ as

$$u_{m,n} = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \widehat{u}_{k,l} \exp\left(\frac{2i\pi mk}{M}\right) \exp\left(\frac{2i\pi nl}{N}\right) \tag{10}$$

for all $0 \leq m \leq M-1$ and $0 \leq n \leq N-1$.

The normalized bi-dimensional Gaussian function of standard deviation σ is defined by

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Its Fourier Transform is

$$\widehat{G}_\sigma(\mu,\nu) = \exp\left(-\sigma^2 \frac{\mu^2+\nu^2}{2}\right)$$

5.2 Exact Gaussian Convolution of an image

Given a digital image $(u_{m,n})$, $0 \leq m \leq M-1$, $0 \leq n \leq N-1$, and interpreting each $u_{m,n}$ as a sampled value on the grid $G = \{(m,n),\ 0 \leq m \leq M-1,\ 0 \leq n \leq N-1\}$, the trigonometric polynomial

$$u(x,y) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \widehat{u}_{k,l} \exp\left(\frac{2i\pi kx}{M}\right) \exp\left(\frac{2i\pi ly}{N}\right)$$

is the only trigonometric polynomial which interpolates the samples $(u_{m,n})_{m,n}$ on the grid G. Here, u is a continuous band limited representation of the image. This representation allows us to apply exactly the convolution of the image with a Gaussian. It is easy to see [19] that the convolution of a pure wave of frequency ξ, $\exp(i\xi x)$ with a kernel f simply is

$$f * \exp(i\xi x) = \widehat{f}(\xi) \exp(i\xi x).$$

As a consequence the convolution of a trigonometric polynomial with a Gaussian kernel is still a trigonometric polynomial with same frequencies and weighted coefficients. More precisely, we have

$$G_\sigma * u = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \widehat{u}_{k,l} G_\sigma * \exp\left(i\frac{2\pi k}{M}x + i\frac{2\pi l}{N}y\right) =$$

$$= \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \left(\widehat{G}_\sigma\left(\frac{2\pi k}{M}, \frac{2\pi l}{N}\right) \widehat{u}_{k,l}\right) \exp\left(i\frac{2\pi k}{M}x + i\frac{2\pi l}{N}y\right)$$

In the above expression the DFT coefficients of the convolved image are weighted by the factor $\widehat{G}_\sigma\left(\frac{2\pi k}{M}, \frac{2\pi l}{N}\right)$, the continuous Fourier transform of the Gaussian function evaluated on the frequency

$\left(\frac{2\pi k}{M}, \frac{2\pi l}{N}\right)$. These new coefficients correspond to a new DFT coefficients of a real valued image. Then, the IDFT of these coefficients is the result of the convolution with the Gaussian function.

The interpolate of a digital image is a function which is defined on the whole plane $\mathbb{R}^2$, since it is periodic. Since it extends the image to the entire plane via periodization, this kind of extension produces discontinuities at the borders of the image. A trick to avoid this edge effect is to first extend the original image $M \times N$ into an even $2M \times 2N$ image and then to convolve this extended image.

5.3 The Algorithm

The proposed algorithm for the exact convolution with a Gaussian follows from the above considerations.

Algorithm 3: Algorithm implementation of an exact Gaussian convolution

Data: σ standard deviation of the Gaussian kernel, monochromatic image u size $M \times N$
Result: The convolved image v.
begin
  Quadruplicate by symmetrization the image $u \to U$
  FFT of $u \to \hat{U}$
  Multiply $\hat{U}$ by the FT of the Gaussian on $\left(\frac{2\pi k}{2M}, \frac{2\pi l}{2N}\right) \to \hat{V}$
  IFFT of $\hat{V} \to V$.
  Restrict V to the image domain $\to v$
end

Remark that this algorithm is applied on monochrome images. For RGB images it applies on each color channel. For the Fourier transform we use the library FFTW[2]. We mentioned the quadruplication for pedagogic reasons since the even and real symmetries allow to reduce the $2M \times 2N$ DFT to an $M \times N$ DCT-II. The complexity of the algorithm is $\mathcal{O}(n \log n)$. The CPU time on an Intel(R) Core(TM) i7-3520M, 2,9 Ghz and 8 Gb of RAM is 0.63 seconds for a 700×528 image.

6 Online Demo

An online demo permits to try Multiscale Retinex on any uploaded image. The demo presents two different versions of the Multiscale Retinex. The first one is the Multiscale Retinex on each color channel, that is faithful to the original work of Jobson et al. [7], except for the final gain/offset step, for which we propose a simplest color balance with a fixed percentage of saturation at either side. The second one is the result of the Multiscale Retinex applied to the intensity channel.

There are five parameters in this algorithm: the three scales, and the percentage of saturation on each side of the histogram, for the simplest color balance.

7 Results

This section presents results of Multiscale Retinex with the final color restoration step. Various options for the final postprocessing step are considered. Results when applying the algorithm on the intensity channel only are compared with results of applying it on each color band, as in the original work.

[2]M. Frigo and S.G. Johnson, *FFTW Package*, http://fftw.org

7.1 Comparison of Different Gain/Offset

First we present several experiments with varying final gain/offset to illustrate how important this operation is for the appearance of the final result. Some of the contrast enhancement is due to this part of the process.

Jobson et al. [8] observed that is possible to find a "canonical gain/offset" which can be applied to all images. We have observed that a gain $G = 30$ and an offset $b = -6$ produces decent results on most images. Nevertheless for others it is better to change the values of the gain/offset. Figure 5 shows an example of MSRCR result with varying gain/offset. They can be are compared with the results obtained by our two versions of the algorithm.

We compared three ways to compute the gain/offset: the "canonical gain/offset", the method used in the plug-in of the Gimp[3] implementation of retinex and our proposition, the simplest color balance [14]. The three methods clip some percentage of the range, but in different ways and with different results. With the "canonical gain/offset" we don't have any information of the percentage of clipped pixels. The Gimp plug-in adjusts the color dynamics based on statistics of the first and second order. The colors are then dispersed around the average, and, the variance allows to control the amount of color saturation. The simplest color balance computes the clipping points for each channel and applies a different gain/offset for each color channel. Figure 6 compares the results with the three different methods.

7.2 MSRCR vs MSRCP

This section compares the results of MSRCR applied on each color channel independently to the results when applying it on the intensity channel only. Applying MSRCP obviously preserves better the image chromaticity. We can conclude that, for scenes with colored illumination, it is better to apply the algorithm on each color channel independently, so that a color balance is also effectuated by MSRCR. On the other hand, for images with a correct color distribution and white lightning, the result of applying MSRCP better preserves the color distribution.

A good illustration of this fact is given by Figure 7 showing an underwater image. This image presents a greenish lightning all over the image. MSRCP applied to the intensity image enhances the greenish aspect, while when applied on RGB the original colors are restored. This color balance effect simulates color perception which is one of the goals of the retinex theory.

Figure 8 shows another example where the result on the intensity channel better preserves the original colors. The result when applying MSRCR becomes grayish.

7.3 MSRCR on Well Contrasted Images

We evaluated the results of the MSRCR algorithm and MSRCP algorithm applied to well-contrasted images namely images where the color balance is correct and where shadow zones are not present. We can observe these results in Figure 9 and Figure 10.

In Figure 9, the results with both methods are not satisfactory. As we have mentioned, the color restoration step may lead to wrong colors. Observe this effect on the yellow chest of the parrot in Figure 9 bottom left, where the saturated yellow is changed to white. Applying the algorithm on the intensity channel yields saturated colors, which do not need modifications.

The results obtained in Figure 10 are notably better. The original image is slightly dark, and this dark tone disappears after applying MSRCR algorithm. Nevertheless observe that the pink color of the flower is faded (Figure 10 bottom left). On the other hand the result of applying MSRCP saturates the colors excessively (Figure 10 bottom right).

[3] *F. Pélisson. Algorithme MSRCR : normalisation couleur, http://www-prima.inrialpes.fr/pelisson/MSRCR.php.*

Figure 5: Comparison of different gain/offset. Top Left: original image. Top right: result of the original MSRCR with $G = 30$ and $b = -6$. Center left: result with $G = 25$ and $b = -6$. Center right: result with $G = 20$ and $b = -10$. Bottom left: result with our version using the simplest color balance, with 1% of saturation, as postprocessing, that is, result of MSRCR algorithm. Bottom right: result with simplest color balance, with 1% of saturation, on the intensity channel, that is, result of MSRCP algorithm.

Acknowledgments

The authors were supported by the Ministerio de Ciencia e Innovación under Grant TIN2011-27539, by MISS project of Centre National d'Etudes Spatiales, the Office of Naval Research under Grant

Figure 6: Comparison of three ways to compute the gain/offse.Top left: original image. Top right: result with $G = 30$ and $b = -6$. Bottom left: Gimp result. Bottom right: result with simplest color balance with 1% of saturation in each channel as postprocessing.

N00014-97-1-0839 and by the European Research Council, advanced grant "Twelve labours".

Image Credits

  Catalina Sbert CC-BY.

 A.B. Petro CC-BY.

 J.L. Lisani CC-BY.

  NASA.[4]

  Kodak image suite.

[4] http://dragon.larc.nasa.gov/retinex/

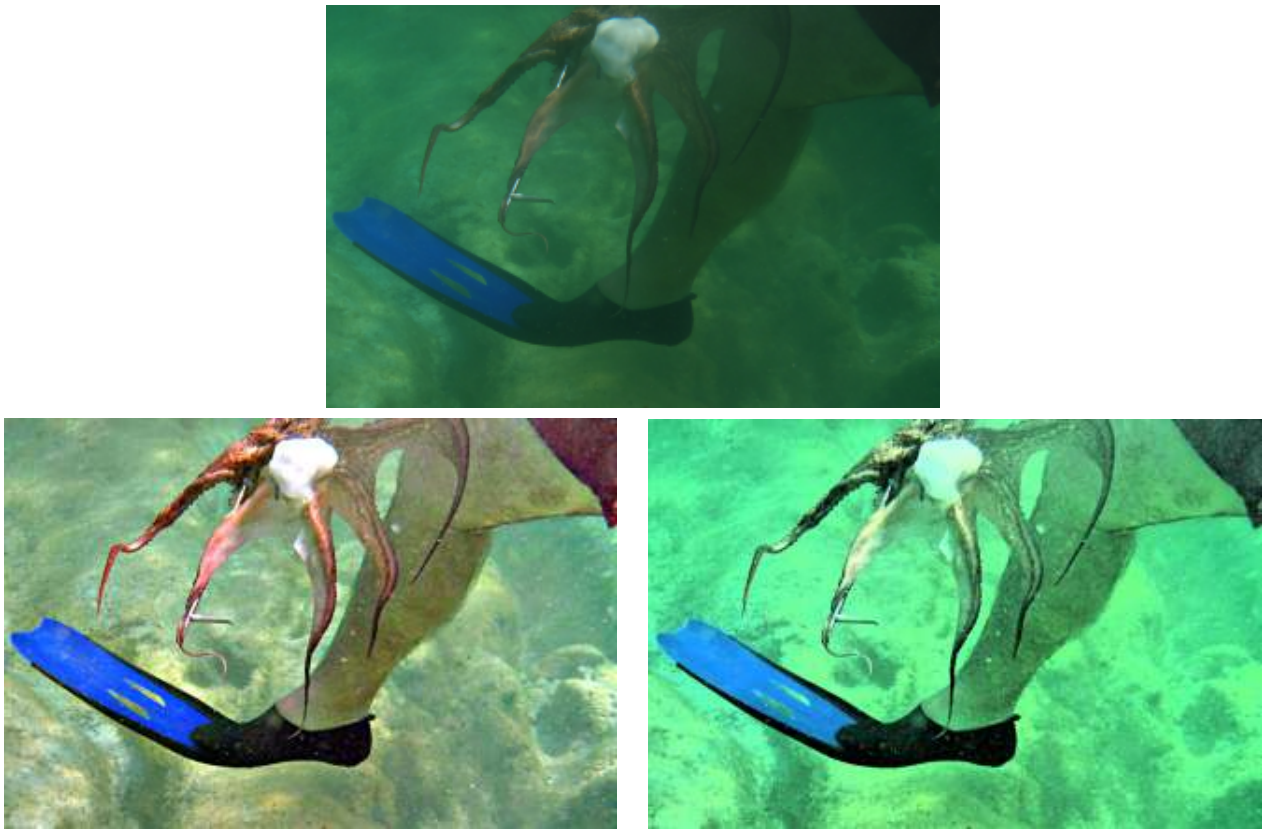

Figure 7: Top: original image. Bottom left: Result of MSRCR. Bottom right: Result of MSRCP.

References

[1] K. Barnard and B. Funt, *Investigations into multi-scale retinex*, Colour Imaging: Vision and Technology, (1999), pp. 9–17. ISBN: 978-0-471-98531-0.

[2] M. Bertalmio, V. Caselles, and E. Provenzi, *Issues about retinex theory and contrast enhancement*, International Journal of Computer Vision, 83 (2009), pp. 101–119. http://dx.doi.org/10.1007/s11263-009-0221-5.

[3] G. Buchsbaum, *A spatial processor model for object colour perception*, Journal of the Franklin institute, 310 (1980), pp. 1–26. http://dx.doi.org/10.1016/0016-0032(80)90058-7.

[4] M. Ebner, *Color constancy*, vol. 6, John Wiley & Sons, 2007. ISBN: 978-0-470-05829-9.

[5] J. Frankle and J. McCann, *Method and apparatus for lightness imaging*, May 17 1983. US Patent 4,384,336.

[6] A. Hurlbert, *Formal connections between lightness algorithms*, Journal of the Optical Society of America A, 3 (1986), pp. 1684–1693. http://dx.doi.org/10.1364/JOSAA.3.001684.

[7] D.J. Jobson, Z. Rahman, and G.A. Woodell, *A multiscale retinex for bridging the gap between color images and the human observation of scenes*, IEEE Transactions on Image Processing, 6 (1997), pp. 965–976. http://dx.doi.org/10.1109/83.597272.

[8] ——, *Properties and performance of a center/surround retinex*, IEEE Transactions on Image Processing, 6 (1997), pp. 451–462. http://dx.doi.org/10.1109/83.557356.

Figure 8: Top: original image. Bottom left: Result of MSRCR. Bottom right: Result of MSRCP.

[9] R. Kimmel, M. Elad, D. Shaked, R. Keshet, and I. Sobel, *A variational framework for retinex*, International Journal of Computer Vision, 52 (2003), pp. 7–23. http://dx.doi.org/10.1023/A:1022314423998.

[10] E. Land, *The retinex theory of color vision*, Scientific American, 237 (1977), pp. 108–128. http://dx.doi.org/10.1038/scientificamerican1277-108.

[11] ——, *Recent advances in retinex theory and some implications for cortical computations: Color vision and the natural image*, Proceedings of the National Academy of Sciences of the United States of America, 80 (1983), pp. 5163–5169. http://dx.doi.org/10.1073/pnas.80.16.5163.

[12] E. Land, *Recent advances in retinex theory*, Vision Research, 26 (1986), pp. 7–21. http://linkinghub.elsevier.com/retrieve/pii/0042698986900672.

[13] E. Land and J. McCann, *Lightness and retinex theory*, Journal of the Optical Society of America, 61 (1971), pp. 1–11. http://dx.doi.org/10.1364/JOSA.61.000001.

[14] N. Limare, J.L. Lisani, J.M. Morel, A.B. Petro, and C. Sbert, *Simplest Color Balance*, Image Processing On Line, 1 (2011). http://dx.doi.org/10.5201/ipol.2011.llmps-scb.

[15] W. Ma, J.M. Morel, S. Osher, and A. Chien, *An L1-based variational model for retinex theory and its application to medical images*, in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2011, pp. 153–160. http://dx.doi.org/10.1109/CVPR.2011.5995422.

Figure 9: Top: Original image. Bottom left: Result of MSRCR. Bottom right: Result of MSRCP.

[16] D. Marini and A. Rizzi, *A computational approach to color adaptation effects*, Image and Vision Computing, 18 (2000), pp. 1005–1014. http://dx.doi.org/10.1016/S0262-8856(00)00037-8.

[17] A. Moore, J. Allman, and R. Goodman, *A real-time neural system for color constancy*, IEEE Transactions on Neural Networks, 2 (1991), pp. 237–247. http://dx.doi.org/10.1109/72.80334.

[18] J.M. Morel, A.B. Petro, and C. Sbert, *A PDE formalization of retinex theory*, IEEE Transactions on Image Processing, 19 (2010), pp. 2825–2837. http://dx.doi.org/10.1109/TIP.2010.2049239.

[19] J.M. Morel and G. Yu, *Is SIFT scale invariant?*, Inverse Problems and Imaging, 5 (2011), pp. 115–136. http://dx.doi.org/10.3934/ipi.2011.5.115.

[20] E. Provenzi, L. D. Carli, A. Rizzi, and D. Marini, *Mathematical definition and analysis of the retinex algorithm*, Journal of the Optical Society of America A, 22 (2005), pp. 2613–2621. http://dx.doi.org/10.1364/JOSAA.22.002613.

[21] E. Provenzi, M. Fierro, A. Rizzi, L. De Carli, D. Gadia, and D. Marini, *Random spray retinex: A new retinex implementation to investigate the local properties of the model*, IEEE Transactions on Image Processing, 16 (2007), pp. 162–171. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4032825.

Figure 10: Top: Original image. Bottom left: Result of MSRCR. Bottom right: Result of MSRCP.